AMENDMENTS TO THE CLAIMS

1. (Currently Amended) A computer-implemented method for loading an object class into computer memory for access by an object-oriented application, comprising the steps of:

loading into the computer memory a first object class, wherein a classload request for the first object class traverses from a first classloader to a second classloader;

wherein a class loading operation accesses the second classloader;

creating a destructible classloader that contains a second object class,

wherein after creating the destructible classloader, the classload request traverses from the first classloader to the destructible classloader and then to the second classloader; ~~and~~

wherein after creating the destructible classloader, the destructible classloader is accessed when the class loading operation with respect to the second classloader is utilized;

wherein the destructible classloader attempts to load into the computer memory the second object class if the second classloader fails to load the second object class.

2. (Original) The method of claim 1 wherein the second object class of the destructible classloader contains a variation of the classes specified in the first object class.

3. (Original) The method of claim 1 further comprising the steps of:

unloading the destructible classloader;

creating a second destructible classloader that contains a third object class,

wherein after creating the second destructible classloader, a classload request for the third object class traverses from the first classloader to the second destructible classloader and then to the second classloader; and

wherein the second destructible classloader attempts to load into the computer memory the third object class if the second classloader fails to load the third object class.

4. (Original) The method of claim 3 wherein the third object class of the second destructible classloader contains a variation of the classes specified in the second object class.

5. (Original) The method of claim 1 wherein the object-oriented application is an object-oriented development environment.

6. (Original) The method of claim 5 wherein the object-oriented development application is operating when the second object class is made accessible to the object-oriented development application by the destructible classloader.

7. (Original) The method of claim 1 wherein the object-oriented application operates substantially throughout a twenty-four hour period, wherein the object-oriented application is operating when the second object class is made accessible to the object-oriented development application by the destructible classloader.

8. (Original) The method of claim 1 wherein the object-oriented application includes web server means.

9. (Original) The method of claim 1 wherein the first classloader is a classloader that originated the classload request.

10. (Original) The method of claim 9 wherein the second classloader is a system classloader.

11. (Original) The method of claim 10 wherein the first and second classloaders operate within a Java-based environment.

12. (Original) The method of claim 1, wherein classload requests traverse a hierarchy of classloaders,

wherein before creating the destructible classloader, the classload requests travel from a user classloader to a system classloader to an extensions classloader to a bootstrap classloader; and

wherein after creating the destructible classloader, the classload requests travel from the user classloader to the destructible classloader to the system classloader to the extensions classloader to the bootstrap classloader.

13. (Original) The method of claim 1, wherein the first and second classloaders operate within a Java-based environment, wherein invocation of a predetermined method returns the second classloader, said method further comprising the step of:

manipulating the method such that the method returns the destructible classloader instead of the second classloader.

14. (Original) The method of claim 13 wherein the destructible classloader is a child of the second classloader.

15. (Original) The method of claim 14 wherein the second classloader is system classloader.

16. (Original) A computer-implemented system for loading object classes into computer memory for access by an object-oriented application, comprising:

a pre-existing class loading hierarchy that specifies parent-child relationships of classloaders;

a class loading operation that provides one of the classloaders in the pre-existing class loading hierarchy when the class loading operation is called, wherein the provided classloader is used to make accessible to the application an object class; and

a classloader switching module that inserts a first destructible classloader into the pre-existing class loading hierarchy by causing the class loading operation to provide the first destructible classloader when the class loading operation is called,

wherein after the first destructible classloader is inserted into the pre-existing class loading hierarchy, the first destructible classloader makes accessible for loading into the computer memory a first object class.

17. (Original) The system of claim 16 wherein the first destructible classloader contains the first object class.

18. (Currently Amended) The system of claim 16 wherein the classloader switching module inserts the first destructible classloader into the pre-existing class loading hierarchy upon receiving a first classloader switch command;

wherein the first destructible classloader is returned as the parent of a user classloader instead of a system classloader; wherein the system classloader is the parent of the first destructible classloader.

19. (Original) The system of claim 16 wherein the classloader switching module unloads the first destructible classloader and loads a second destructible classloader upon receiving a second classloader switch command, wherein the second destructible classloader makes accessible for loading into the computer memory a second object class.

20. (Original) The system of claim 19 wherein the second object class includes classes that were not in the first object class.

21. (Original) The system of claim 19 wherein the second object class includes classes that are a variation of at least one class in the first object class.

22. (Original) The system of claim 16 wherein the object-oriented application is an object-oriented development environment.

23. (Original) The system of claim 22 wherein the object-oriented development application is operating when the first object class is made accessible to the object-oriented development application by the first destructible classloader.

24. (Original) The system of claim 16 wherein the object-oriented application operates substantially throughout a twenty-four hour period, wherein the object-oriented application is operating when the first object class is made accessible to the object-oriented development application by the first destructible classloader.

25. (Original) The system of claim 16 wherein the object-oriented application includes web server means.

26. (Original) The system of claim 16 wherein the first destructible classloader operates within a Java environment, wherein the pre-existing class loading hierarchy includes a system classloader, wherein the classloader switching module manipulates the class loading operation such that the class loading operation returns the first destructible classloader instead of the system classloader.

27. (Original) The system of claim 26 wherein the first destructible classloader is a child of the system classloader.

28. (Original) The system of claim 27 wherein the class loading operation is an object-oriented based method call within the Java environment.